

**Search Forms**  
**Search Results**

# Freeform Search

**Help**

**User Searches**

**Preferences**

**Logout**

Database:

US Pre-Grant Publication Full-Text Database  
 US Patents Full-Text Database  
 US OCR Full-Text Database  
 EBO Abstracts Database  
 JPO Abstracts Database  
 Derwent World Patents Index  
 IBM Technical Disclosure Bulletins

Term:

Display:  Documents in Display Format:  Starting with Number

Generate: ☐ Hit List ☒ Hit Count ☐ Side by Side ☐ Image

## Search History

DATE: Thursday, April 28, 2005 [Printable Copy](#) [Create Case](#)

<u>Set</u> <u>Name</u>	<u>Query</u>	<u>Hit</u> <u>Count</u>	<u>Set</u> <u>Name</u> result set
side by side			
	DB=USPT; PLUR=YES; OP=OR		
<u>L30</u>	5854933.pn.	1	<u>L30</u>
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
<u>L29</u>	L27 and (java or c or c++ or fortran or cobol)	33	<u>L29</u>
<u>L28</u>	L27 and macro	0	<u>L28</u>
<u>L27</u>	L25 and uncompiled near code	33	<u>L27</u>
<u>L26</u>	L25 and (uncompiled or uncompiled\$) near program with code	15	<u>L26</u>
<u>L25</u>	(pre-processor near macro or preprocessor near macro or runtime or run adj time or run with time or run-time)	161109	<u>L25</u>
<u>L24</u>	(pre-processor near macro or preprocessor near macro)	114	<u>L24</u>
<u>L23</u>	L22 and (extend or expand)	114	<u>L23</u>
<u>L22</u>	19 and (macro near language or macro with language)	225	<u>L22</u>
<u>L21</u>	715/209	0	<u>L21</u>
<u>L20</u>	707/209	7	<u>L20</u>
<u>L19</u>	717/140	439	<u>L19</u>
<u>L18</u>	717/124	605	<u>L18</u>

<u>L17</u>	717/122	152	<u>L17</u>
<u>L16</u>	717/118	234	<u>L16</u>
<u>L15</u>	717/117	77	<u>L15</u>
<u>L14</u>	717/116	440	<u>L14</u>
<u>L13</u>	717/115	128	<u>L13</u>
<u>L12</u>	717/114	314	<u>L12</u>
<u>L11</u>	717/106	327	<u>L11</u>
<u>L10</u>	717/209	0	<u>L10</u>
<u>L9</u>	717.clas.	8349	<u>L9</u>
<u>L8</u>	715/526	278	<u>L8</u>
<u>L7</u>	715/513	1182	<u>L7</u>
<u>L6</u>	715/500	548	<u>L6</u>
<u>L5</u>	715.clas.	21180	<u>L5</u>
<u>L4</u>	707.clas.	26372	<u>L4</u>
<u>L3</u>	707/100	5967	<u>L3</u>
<u>L2</u>	707/2	4337	<u>L2</u>
<u>L1</u>	707/1	6637	<u>L1</u>

END OF SEARCH HISTORY

[First Hit](#) [Fwd Refs](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)[Search Forms](#)[End of Result Set](#)[Search Results](#)

Generate Collection

Print

[Help](#)[User Searches](#)[Preferences](#)

Entry 114 of 114

File: USPT

Feb 14, 1984

[Logout](#)

US-PAT-NO: 4432051

DOCUMENT-IDENTIFIER: US 4432051 A

TITLE: Process execution time accounting system

DATE-ISSUED: February 14, 1984

## INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bogaert; Jean-Louis	Clamart			FR
deRivet; Philippe-Hubert	Paris			FR
Franklin; Benjamin S.	Cambridge	MA		

## ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Honeywell Information Systems Inc.	Waltham	MA			02

APPL-NO: 06/ 202423 [\[PALM\]](#)

DATE FILED: October 30, 1980

## PARENT-CASE:

This application is a continuation of U.S. patent application Ser. No. 055,200 filed Jan. 22, 1979 and now abandoned, which was a continuation of U.S. patent application Ser. No. 823,315, filed Aug. 10, 1977 and now abandoned, which was a continuation of U.S. patent application Ser. No. 529,016, filed Dec. 2, 1974, and now abandoned.

## FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
FR	73 42702	November 30, 1973

INT-CL: [03] G06F 3/00, G06F 9/22

US-CL-ISSUED: 364/200

US-CL-CURRENT: [717/127](#); [718/102](#)

FIELD-OF-SEARCH: 364/2MSFile, 364/9MSFile

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected

Search ALL

Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>3639912</u>	February 1972	Campbell	364/200
<input type="checkbox"/>	<u>3723975</u>	March 1973	Kurtz, Jr. et al.	364/200
<input type="checkbox"/>	<u>3771144</u>	November 1973	Belady et al.	364/200
<input type="checkbox"/>	<u>3818458</u>	June 1974	Deese	364/200

ART-UNIT: 232

PRIMARY-EXAMINER: Springborn; Harvey E.

ATTY-AGENT-FIRM: Grayson; George Prasinos; Nicholas Gaybrick; Robert J.

## ABSTRACT:

A time accounting system for accounting for the time a process spends in a ready state, a wait state, or a running state. The system includes a time-of-day clock coupled to a central processing unit for outputting the time-of-day whenever a process changes state. A memory also coupled to the central processing unit stores the contents of a plurality of addressable process control blocks and each process control block includes first, second, and third storage locations for storing indications of the amount of time an associated process has been in the running, ready, and wait states, respectively. The central processor unit accesses the process control blocks and updates the process times stored therein in accordance with control signals generated by decoding a string of microinstructions stored in a control store memory. The time-of-day clock is accessed each time a process enters one of the running, ready, or wait states and each time the execution of a process is completed. The process control blocks are accessed and updated with the differences in the access times of the time-of-day clock to accumulate in the appropriate storage locations in each process control block the amount of time the associated process was in the running, ready, or wait states.

4 Claims, 57 Drawing figures

[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

[First Hit](#) [Fwd Refs](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)[Search Forms](#)[Generate Collection](#)[Print](#)[Search Results](#)[Help](#)[User Searches](#)

123: Entry 112 of 114

File: USPT

Apr 10, 1990

[Preferences](#)[Logout](#) NO: 4916610

DOCUMENT-IDENTIFIER: US 4916610 A

TITLE: Multilanguage software integration through preprocessing

DATE-ISSUED: April 10, 1990

## INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bapat; Subodh	Fort Lauderdale	FL		

## ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Racal Data Communications Inc.	Sunrise	FL			02

APPL-NO: 07/ 253464 [\[PALM\]](#)

DATE FILED: October 5, 1988

INT-CL: [04] G06F 9/00

US-CL-ISSUED: 364/300

US-CL-CURRENT: [717/141](#); [717/114](#), [717/139](#), [717/146](#)

FIELD-OF-SEARCH: 364/200, 364/300, 364/900

PRIOR-ART-DISCLOSED:

## U.S. PATENT DOCUMENTS

[Search Selected](#)[Search ALL](#)[Clear](#)

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<a href="#">4309756</a>	January 1982	Beckler	364/300
<input type="checkbox"/>	<a href="#">4315315</a>	February 1982	Kossiakoff	364/300
<input type="checkbox"/>	<a href="#">4330822</a>	May 1982	Dodson	364/200
<input type="checkbox"/>	<a href="#">4374408</a>	February 1983	Bowles et al.	364/200
<input type="checkbox"/>	<a href="#">4414629</a>	November 1983	Waite	364/300
<input type="checkbox"/>	<a href="#">4463423</a>	July 1984	Potash et al.	364/300
<input type="checkbox"/>	<a href="#">4559614</a>	December 1985	Peek et al.	364/900
<input type="checkbox"/>	<a href="#">4566078</a>	January 1986	Crabtree	364/900
	<a href="#">4567574</a>	January 1986	Saade et al.	364/900

<input type="checkbox"/>				
<input type="checkbox"/>	<u>4595981</u>	June 1986	Leung	364/300
<input type="checkbox"/>	<u>4692896</u>	September 1987	Sakoda et al.	364/900
<input type="checkbox"/>	<u>4719564</u>	January 1988	Hara	364/200

## OTHER PUBLICATIONS

Advanced Turbo Pascal Programming & Techniques, Herbert Schildt, 1986, McGraw Hill, pp. 108-124.

CTIX Operating System Manual, Convergent Technologies, date unknown, pp. 1-2.

The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie, 1978, Prentice-Hall, Inc., pp. 86-87 and 207-208.

ART-UNIT: 232

PRIMARY-EXAMINER: Zache; Raulfe B.

ATTY-AGENT-FIRM: Miller; Jerry A.

## ABSTRACT:

A method of assuring consistency of constants in a multilanguage software system, includes generating a first set of code written in a first language using a plurality of symbolic constants to represent a corresponding plurality of actual constants. A second set of code is written in a second language using the plurality of symbolic constants to represent the corresponding plurality of actual constants. A common header file is generated which contains information which relates the plurality of symbolic constants to the corresponding plurality of actual constants. The header is included within the first and second sets of code. The symbolic constants in the first and second sets of code are replaced with their corresponding actual constants during a preprocessing step. Any constructs which are not a part of the first language are stripped from the second set of code including the header file. Any constructs which are not a part of the second language are stripped from the first set of code including the header. The resulting files have their symbolic constants consistently replaced by constants which are defined in the header so that changes need only be made in the header file to assure consistency in the several languages.

11 Claims, 1 Drawing figures

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#) [Fwd Refs](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

End of Result Set



Generate Collection

Print

L29: Entry 33 of 33

File: USPT

Dec 28, 1999

US-PAT-NO: 6009273

DOCUMENT-IDENTIFIER: US 6009273 A

TITLE: Method for conversion of a variable argument routine to a fixed argument routine

DATE-ISSUED: December 28, 1999

## INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ayers; Andrew E.	Amherst	NH		
Liu; Jiyang	Westford	MA		

## ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Hewlett-Packard Company	Palo Alto	CA			02

APPL-NO: 08/ 953549 [PALM]

DATE FILED: October 21, 1997

## PARENT-CASE:

This Application claims priority from U.S. Provisional patent application Ser. No. 60/047,866, filed May 29, 1997.

INT-CL: [06] G06 F 9/44

US-CL-ISSUED: 395/709, 395/708, 395/704, 395/705, 712/226, 712/234, 712/241

US-CL-CURRENT: 717/143, 712/226, 712/234, 712/241, 717/157, 717/160

FIELD-OF-SEARCH: 295/709, 295/708, 295/704, 295/705, 295/567, 295/580, 295/581, 295/588, 712/226, 712/233, 712/234, 712/241

## PRIOR-ART-DISCLOSED:

## U.S. PATENT DOCUMENTS

Search Selected

Search ALL

Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>5740443</u>	April 1998	Carini	395/705
<input type="checkbox"/>	<u>5854933</u>	December 1998	Chang	395/709

## OTHER PUBLICATIONS

Kaser et al., "On the Conversion of Indirect to Direct Recursion," ACM Letters on Programming Languages and Systems, vol. 2, No. 1-4, pp. 151-164, Mar. 1993.  
Frank Vahid, "Procedure Cloning: A Transformation for Improved System-Level Functional Partitioning," Proceedings of European Design and Test Conference, ED&TC 97, pp. 487-492, Mar. 1997.  
Cooper et al., "Procedure Cloning," Proceedings of the 1992 International Conference on Computer Languages, pp. 96-105, Apr. 1992.  
Cooper et al., "Unexpected Side Effects of Inline Substitution: A Case Study," ACM Letters on Programming Languages and Systems, vol. 1, No. 1, pp. 22-32, Mar. 1992.

ART-UNIT: 272

PRIMARY-EXAMINER: Hafiz; Tariq R.

ASSISTANT-EXAMINER: Dam; Tuan Q.

ABSTRACT:

A compiler method analyzes a program listing to identify a first set of subroutines therein, each of which accepts a variable number of arguments, converting the first set of subroutines into further sets of subroutines which accept fixed numbers of arguments. The method includes the steps of: locating subroutines in the program listing which accept a variable number of arguments and identifying which thereof comprise a first set of subroutines that can be altered to a form which accepts a fixed number of arguments; for each subroutine identified as part of the first set, determining call sites which pass arguments to each subroutine and determining a number and kind of said arguments to be passed therefrom; partitioning call sites to each subroutine of the first set into one or more groups, each group comprising call sites which pass an identical number and kind of arguments to an associated subroutine of said first set; duplicating each subroutine of said first set into plural second sets of N corresponding subroutines, wherein N is equal to the number of groups associated with the subroutine of the first set, and revising each one of said N corresponding subroutines to receive a number of fixed arguments to be passed by call sites partitioned into an associated group; and substituting in the program listing the N corresponding subroutines and revising call sites in each group to refer to an associated one of said N corresponding subroutines.

8 Claims, 5 Drawing figures

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)



[First Hit](#) [Fwd Refs](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)[Search Forms](#)

Generate Collection

Print

[Search Results](#)[Help](#)[User Searches](#)

L24: Entry 97 of 114

File: USPT

Oct 21, 1997

[Preferences](#)

USPT NO: 5680585

DOCUMENT-IDENTIFIER: US 5680585 A

TITLE: Method and apparatus for defining data packet formats

DATE-ISSUED: October 21, 1997

## INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bruell; Gregory O.	Carlisle	MA		

## ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Bay Networks, Inc.	Santa Clara	CA			02

APPL-NO: 08/ 414445 [\[PALM\]](#)

DATE FILED: March 31, 1995

INT-CL: [06] [G06 F 13/00](#), [G06 F 13/42](#), [G06 F 15/00](#)

US-CL-ISSUED: 395/500; 364/240.8, 364/260.1, 364/280.4, 364/284

US-CL-CURRENT: [703/26](#)

FIELD-OF-SEARCH: 395/200.18, 395/200.2, 395/500, 395/375, 395/800

PRIOR-ART-DISCLOSED:

## U.S. PATENT DOCUMENTS

Search Selected

Search ALL

Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<a href="#">5021947</a>	June 1991	Campbell et al.	395/800
<input type="checkbox"/>	<a href="#">5475838</a>	December 1995	Fehskens	395/185.1
<input type="checkbox"/>	<a href="#">5483640</a>	January 1996	Isfeld et al.	395/200.03
<input type="checkbox"/>	<a href="#">5515508</a>	May 1996	Pettus et al.	395/200.01
<input type="checkbox"/>	<a href="#">5524253</a>	June 1996	Pham et al.	395/800

ART-UNIT: 234

PRIMARY-EXAMINER: Teska; Kevin J.

ASSISTANT-EXAMINER: Mohamed; Ayni

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

A packet description language which is declarative in nature and suitable for efficiently and flexibly defining data packet formats in accordance with internetwork routing device uses. Data packet formats may be defined utilizing the packet description language and then compiled to create a data structure corresponding to the defined data packet format. A routing device test platform may generate test data packets and decode received test packets by referencing the test data to the compiled data structure defined in accordance with the packet description language. The declarative language provides for assigning numerous default values and attributes to packet fields such that only a small amount of data need be specified when regression testing a new routing device.

14 Claims, 5 Drawing figures

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#) [Fwd Refs](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

Generate Collection

Print

L24: Entry 102 of 114

File: USPT

Jan 4, 1994

US-PAT-NO: 5276880

DOCUMENT-IDENTIFIER: US 5276880 A

TITLE: Method for parsing and representing multi-versioned computer programs, for simultaneous and synchronous processing of the plural parses

DATE-ISSUED: January 4, 1994

## INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Platoff; Michael A.	Monmouth Junction			NJ
Wagner; Michael E.	Hopewell Township, Mercer County			NJ

## ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Siemens Corporate Research, Inc.	Princeton	NJ			02

APPL-NO: 07/ 451493 [PALM]

DATE FILED: December 15, 1989

INT-CL: [05] G06F 9/45

US-CL-ISSUED: 395/700; 364/280, 364/280.4, 364/280.5, 364/DIG.1

US-CL-CURRENT: 717/143; 717/154

FIELD-OF-SEARCH: 364/DIG.1, 364/DIG.2, 395/700

PRIOR-ART-DISCLOSED:

## U.S. PATENT DOCUMENTS

Search Selected

Search ALL

Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>5050068</u>	September 1991	Dollas et al.	395/375
<input type="checkbox"/>	<u>5105353</u>	April 1992	Charles et al.	395/700
<input type="checkbox"/>	<u>5151991</u>	September 1992	Iwasawa et al.	395/700

## FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0343883	November 1989	EP	

## OTHER PUBLICATIONS

"The Theory and Practice of Compiler Writing", Tremblay et al., McGraw-Hill Book Company, pp. 1-13.

"Using Tomorrow's C Standard Today", Williams et al., Computer Language, Jul. 1988, vol. 5, pp. 49-54.

"A Coroutine Approach to Parsing", Hanan Samet, ACM Transactions on Programming Languages and Systems, vol. 2, No. 3, Jul. 1980, pp. 290-306.

Patent Abstracts of Japan, vol. 9, No. 214 (P-384) Aug. 31, 1985 & JP-A-60 074 039 (Fujitsu K. K.) Apr. 26, 1985.

"Bilingual Mapping Macro", T. L. Oliver, IBM Technical Disclosure Bulletin, vol. 13, No. 10, Mar. 1971, p. 2903.

ART-UNIT: 237

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Von Buhr; Maria N.

ATTY-AGENT-FIRM: Ahmed; Adel A.

## ABSTRACT:

A modified parser generator enables the parsing of programs such as C programs with preprocessor directives and implemented extensions to the normal abstract syntax tree representation of programs to create an integrated representation structure.

21 Claims, 3 Drawing figures

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#)   [Fwd Refs](#)   [Previous Doc](#)   [Next Doc](#)   [Go to Doc#](#)

☐ [Generate Collection](#)   [Print](#)

L24: Entry 104 of 114

File: USPT

Jun 5, 1990

US-PAT-NO: 4931928

DOCUMENT-IDENTIFIER: US 4931928 A

**\*\* See image for Certificate of Correction \*\***

TITLE: Apparatus for analyzing source code

DATE-ISSUED: June 5, 1990

## INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Greenfeld; Norton R.	Wayland	MA	01778	

APPL-NO: 07/ 269135   [\[PALM\]](#)

DATE FILED: November 9, 1988

INT-CL: [05] G06F 9/44

US-CL-ISSUED: 364/300

US-CL-CURRENT: [717/131](#); [707/3](#), [717/114](#), [717/142](#), [717/143](#)

FIELD-OF-SEARCH: 364/200, 364/300, 364/900

PRIOR-ART-DISCLOSED:

## U.S. PATENT DOCUMENTS

[Search Selected](#)[Search ALL](#)[Clear](#)

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<a href="#">4506326</a>	March 1985	Shaw et al.	364/300
<input type="checkbox"/>	<a href="#">4686623</a>	August 1987	Wallace	364/300
<input type="checkbox"/>	<a href="#">4688195</a>	August 1987	Thompson et al.	364/300
<input type="checkbox"/>	<a href="#">4729096</a>	March 1988	Larson	364/300
<input type="checkbox"/>	<a href="#">4751635</a>	June 1988	Kret	364/200
<input type="checkbox"/>	<a href="#">4787035</a>	November 1988	Bourne	364/300

## OTHER PUBLICATIONS

"Global Program Analysis In An Interactive Environment" by Larry M. Masinter, SSL-80-1 Xerox Palo Alto Research Center, Palo Alto, Calif., Jan. 1980, pp. 39-61 and 67-83 (Chapter 4-6 and Appendixes 1-3 respectively).

"Telescope: A Cross Reference Utility for Lisp" by Jed Krohnfeldt, Utah PASS Project Op Note 86-11, Dec. 4, 1986, pp. 1-2.

"Reverserver: Databases for Reverse Engineering" in Release 1.0, published by EDventure Holding, Inc., Apr. 10, 1989, pp. 12-13.

ART-UNIT: 232

PRIMARY-EXAMINER: Zache; Raulfe B.

ATTY-AGENT-FIRM: Hamilton, Brook, Smith & Reynolds

ABSTRACT:

Apparatus in a computer system provides source code analysis. The apparatus includes an analysis member which extracts programming semantics information from an input source code. The analysis member operates according to the programming language of the source code as defined by a grammar mechanism. The analysis member employs a database interface which enables the extracted programming semantics information to be placed in a user desired database for subsequent recall by a desired query system. The database and query system may be pre-existing elements which are supported by a digital processor independently of the analysis member. A relational database with an SQL query system may be used.

19 Claims, 8 Drawing figures

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#)   [Previous Doc](#)   [Next Doc](#)   [Go to Doc#](#)[Search Forms](#)[Search Results](#)[Help](#)[User Searches](#)

L24: Entry 105 of 114

[Preferences](#)[Generate Collection](#)[Print](#)

File: JPAB

Apr 10, 1998

[Logout](#) JP410091166A

DOCUMENT-IDENTIFIER: JP 10091166 A

TITLE: ACOUSTIC MODEL GENERATION DEVICE AND PROGRAM OPTIMIZING METHOD IN ACOUSTIC MODEL GENERATION DEVICE

PUBN-DATE: April 10, 1998

## INVENTOR-INFORMATION:

NAME

COUNTRY

CHARLES, BRIGHT

FUJIWARA, SHUNGO

SOMA, DAIZABURO

## ASSIGNEE-INFORMATION:

NAME

COUNTRY

KK KORUGU

APPL-NO: JP08266624

APPL-DATE: September 18, 1996

INT-CL (IPC): G10 H 5/04; G10 H 1/02

## ABSTRACT:

PROBLEM TO BE SOLVED: To enable increasing functions of a program by retrenching and reducing needless and redundant programs by optimization processing of a program.

SOLUTION: A processor depending section is constituted with a macro- preprocessor 21 converting a macro-preprocessor input list 20 to a form being able to compile, a compiler 22 translating a processed result of the macro- preprocessor 21 to an acoustic synthesis program being performable, and an optimizer 23 performing optimization processing of a program. Indication of execution is performed in order for the macro-preprocessor 21, the compiler 22, and the optimizer 23 through a lock edition tool 18. And the macro- preprocessor input list 20 is converted to a form being able to compile, and the processed result is compiled in the acoustic synthesis program being performable. Further, processing for reducing needless steps and redundant steps out of the acoustic synthesis program being performable is performed.

COPYRIGHT: (C) 1998, JPO

[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

[First Hit](#)   [Previous Doc](#)   [Next Doc](#)   [Go to Doc#](#)[Search Forms](#)[Search Results](#)[Help](#)[User Searches](#)

124: Entry 106 of 114

File: TDBD

Mar 1, 1995

[Preferences](#)[Logout](#) - NO: NN9503641☐ [Generate Collection](#) [Print](#)

DISCLOSURE TITLE: Automated Support Code Generation for an Object Oriented System

## PUBLICATION-DATA:

IBM Technical Disclosure Bulletin, March 1995, US

VOLUME NUMBER: 38

ISSUE NUMBER: 3

PAGE NUMBER: 641 - 646

PUBLICATION-DATE: March 1, 1995 (19950301)

CROSS REFERENCE: 0018-8689-38-3-641

## DISCLOSURE TEXT:

A method of generating supporting source code based on information extracted from a project's Object Oriented Analysis (OOA) models is disclosed. Code generation can be automated and controlled by an underlying architecture domain to help achieve the desired system performance and flexibility goals while preserving a framework derived from extensive modeling analysis. Excerpts of a functional implementation are used as examples to illustrate the interrelation between application objects and the architecture domain. In addition to basic object characteristics such as object name and object type, other artificially asserted characteristics can be enforced during analysis to improve the modeling representation and design documentation. This facilitates error checking within models and enables automated code generation. The following asserted object characteristics are used for automated code generation: o unique object alias name o requirement for instance list support o requirement for subtype migration support o resident processor o transition control statements o event naming conventions that uniquely define each event o event tracing class Two different types of processing are involved in automated code generation. System wide definitions for data structures, global types, constants, and macros are generated using a model extraction tool set. Object specific definitions and code are generated through subsequent macro preprocessing using a compiler tool set. Given a software tool capable of extracting information from analysis models, a set of local macro preprocessor definitions are generated for each object in a file with a name derived from the object's alias name. These local macro preprocessor definitions define the basic object characteristics and are used to control the generation of specific supporting code during object module compilation. These local macros define the following information in macro language syntax: o object full name o object alias name o object type (Active, Passive, or Monitor) o object storage class and size o object requirement for instance list support o object inheritance level (Supertype, Subtype, or Stand-Alone) o if an object has a supertype, the supertype full name o if an object has a supertype, the supertype alias name Prior to object module compilation, a macro preprocessor is invoked to process the source code files and replace all defined local macros with assigned text strings. This creates specific symbols and code that are only meaningful to a particular object in specific implementation language syntax. As an example, to support a doubly linked instance list for a supertype object, two additional attributes are generated and inserted



into an instance data structure to maintain links between the supertype's instances. To guarantee that these attributes are uniquely defined within an instance data structure, each attribute is prefixed with the specific supertype alias name. The process of generating these attributes uses a single code template for all inserted attributes. An example follows: Prerequisites: local macro: zz\_level (object inheritance level) local macro: zz\_inst\_list (requirement for instance list support) local macro: zz\_alias (object alias name) Process of Code Generation for Insertion of Link Attributes: Description: If this object is a supertype object and it requires instance list support, insert two link attributes prefixed with this object's alias name to its instance data structure definition Pseudo Code Template:

SECURITY: Use, copying and distribution of this data is subject to the restrictions in the Agreement For IBM TDB Database and Related Computer Databases. Unpublished - all rights reserved under the Copyright Laws of the United States. Contains confidential commercial information of IBM exempt from FOIA disclosure per 5 U.S.C. 552(b) (4) and protected under the Trade Secrets Act, 18 U.S.C. 1905.

COPYRIGHT STATEMENT: The text of this article is Copyrighted (c) IBM Corporation 1995. All rights reserved.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#)   [Previous Doc](#)   [Next Doc](#)   [Go to Doc#](#)[Search Forms](#)[Search Results](#)[Help](#)[User Searches](#)

124: Entry 109 of 114

File: TDBD

Jan 1, 1979

[Preferences](#)[Logout](#)-NO: NN79013356

Generate Collection

Print

DISCLOSURE TITLE: General Purpose Assembler For Microprocessors. January 1979.

## PUBLICATION-DATA:

IBM Technical Disclosure Bulletin, January 1979, US

VOLUME NUMBER: 21

ISSUE NUMBER: 8

PAGE NUMBER: 3356 - 3357

PUBLICATION-DATE: January 1, 1979 (19790101)

CROSS REFERENCE: 0018-8689-21-8-3356

## DISCLOSURE TEXT:

2p. Using macros with existing assemblers for microprocessors presents problems in that either a new assembler has to be written for each new microprocessor or the use of any existing assembler is restricted by the architecture of the original machine for which it was designed. This article describes a General Purpose Assembler (GPA) designed for use with a macro preprocessor and which gives flexible bit handling and error message generation. The assembler is capable of general application and is not restricted to byte orientated architecture. It can be used for microprocessors, micro programming and programmable logic arrays (PLAs). - In order to generate an assembler for a given source language using the GPA. the user must write a series of macros to translate the source language into the powerful low level language understood by the GPA. - The principle of using macros to translate one assembly language into another is not new, but hitherto this approach has always led to problems due to differences in the architectures of the new and old languages. A particular major problem is dealing with error conditions unique to each processor, for example, testing whether branches are in range, etc. - The GPA solves this problem by permitting conditional error messages which can test assemble time values (equated symbols, address counter etc). - The figure shows the general structure of the GPA. The source program may include macros processed by the same processor. The language in which the macros are written depends upon the macro processor, most stand-alone macro processors are suitable (ML/1, GMP could be used). - The temporary file holds the original source records, interspersed with lower level GPA statements. (These statements do not appear on the listing.) The GPA is a single program written in PL/1 and is readily transportable. The GPA assembles the bit strings required and carries out the necessary housework to produce a useful listing. - The listing is a list of source code with error messages, cross references and error totals.

SECURITY: Use, copying and distribution of this data is subject to the restrictions in the Agreement For IBM TDB Database and Related Computer Databases. Unpublished - all rights reserved under the Copyright Laws of the United States. Contains confidential commercial information of IBM exempt from FOIA disclosure per 5 U.S.C. 552(b) (4) and protected under the Trade Secrets Act, 18 U.S.C. 1905.

COPYRIGHT STATEMENT: The text of this article is Copyrighted (c) IBM Corporation 1979. All rights reserved.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)